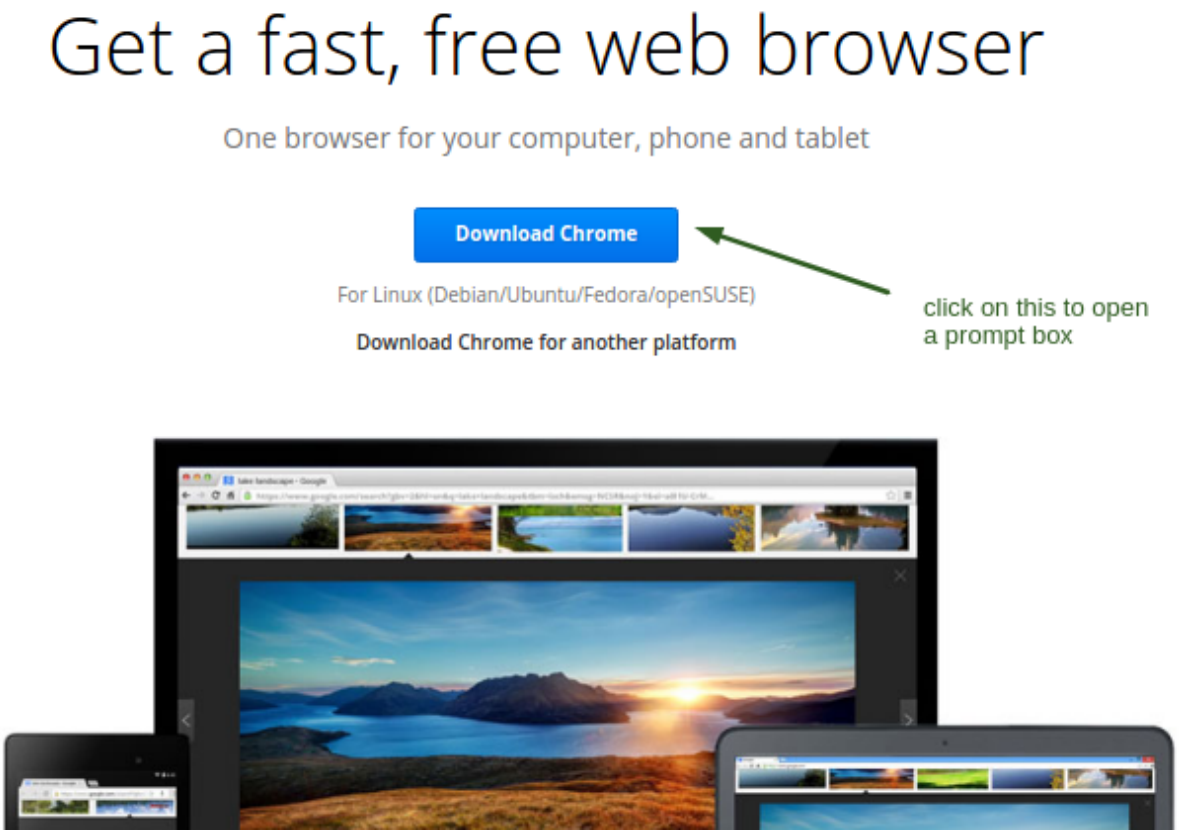
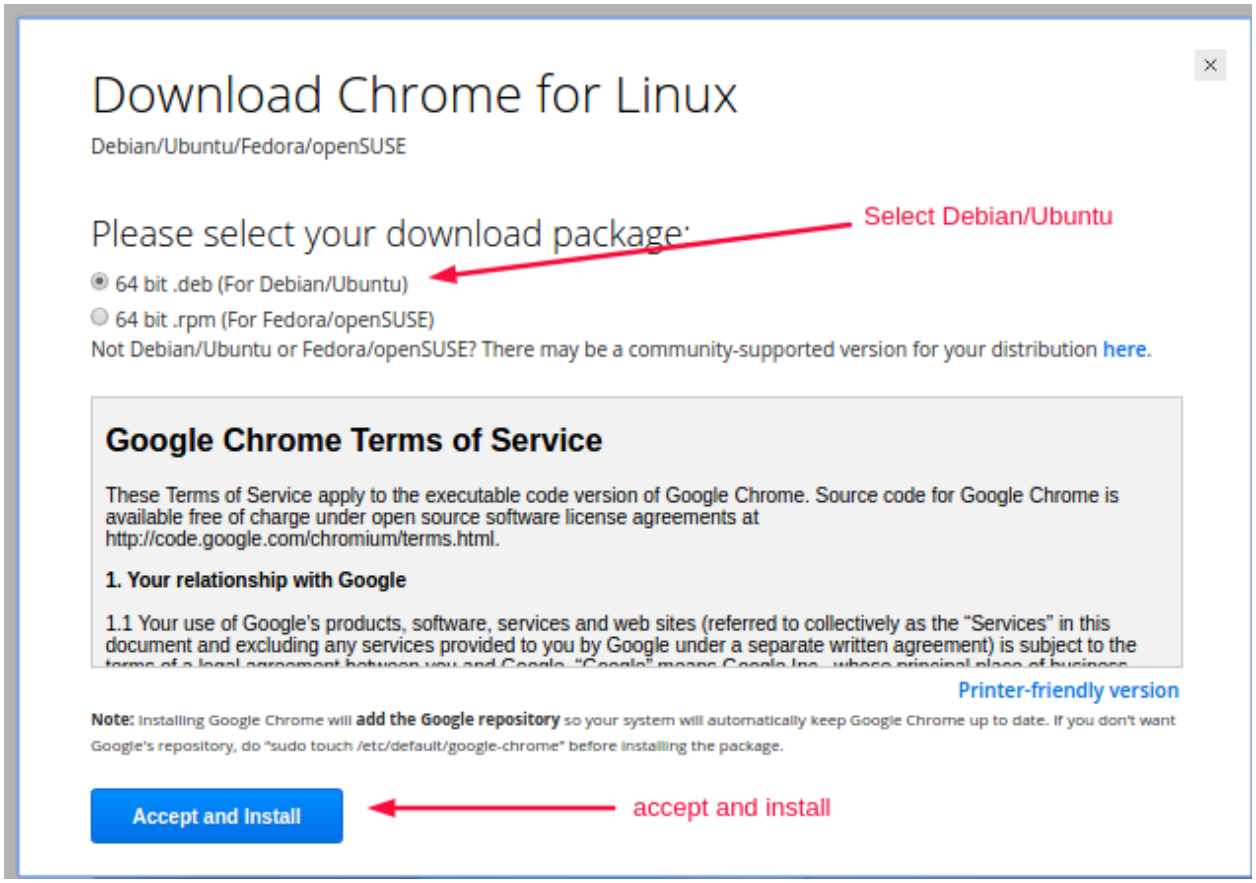
Pre-Course Installations

Google Chrome

At Moringa School, we will typically use **Google Chrome** as the default browser for all of our Data Science work, and our curriculum is written assuming that you are using Chrome. The additional tools built for programmers for Chrome are the best, though you may explore cross-compatibility between web browsers later. If you do not already have Chrome installed, [click this link] and then follow the steps in the images shown below:





Git and GitHub

When we get started writing a new program, the last thing we want to happen is to lose any of our hard work. To prevent that terrible possibility from happening, we will save versions of our code on our computer AND also save a version of our code on a location on the web. The reason why we store our code on the web is to prevent loss of versions that you may have made on your computer. In case your hard drive gets corrupted or worse, you lose your laptop, all you need to do is retrieve your work from the web. Storing our code on the web via Github is also a great avenue for collaboration. When working with other people, you and your teammates can all contribute to the same project. At Moringa School, we will learn to manage these different versions of our code using two tools: **Git** and **GitHub**.

Git is a **version control system** that we install on the computers where we write code. Git allows us take a snapshot of our program, called a **commit**, at important points

along the way as we work. You can think of a commit like a save on a document that you are writing. Although unlike a save, once a commit is made, it is permanently stored in the history of the project. A newer commit does not overwrite the previous commit. If we ever want to go back to the place in time where the commit was made, where all the files looked exactly as they did when we committed, we will be able to do so.

When we are ready to save the code from our computers to a location on the web, we will use GitHub. GitHub allows us to create **repositories** to store the code and code history for each of our projects. A repository is what we call the container that holds all of the files from our project with all of the commits that have been made to it.

Each project that we create will be saved to a repository on GitHub. By the end of your time here, you will have a portfolio of your work represented by all of your GitHub repositories. LinkedIn will be your employment resume and GitHub will serve as your coding resume.

Note that all repositories on GitHub are public by default. For a monthly fee, GitHub offers unlimited private repositories. However, this is entirely optional. Private repos are *not* required for this course.

Installation Instructions: Windows

Head to https://gitforwindows.org/, download and install Git for Windows: Git BASH. Git for Windows provides a BASH emulation used to run Git from the command line.

Installation Instructions: Linux

To use these instructions, you will need to use your computer's command prompt, an application named *Terminal*. The Terminal application on a PC with Linux installed can be accessed by pressing *Ctrl* + *Alt* + *T*. Alternatively, you can locate the Terminal application by searching "Terminal" once you click on the Linux icon on the upper left side of your screen.

We will go much more into detail about how to use the command prompt when we start coding, but for now, simply note that the commands here are preceded by a `$`. When typing the commands yourself, do not type the $ - this is simply a common convention used to indicate that what follows is a command to be run.

Linux has a powerful software installation tool known as Advanced Packaging Tool, which is accessed with the `apt` command. To install Git via apt, run the following commands:

```
$ sudo apt-get update
```

```
$ sudo apt-get install git
```

Next, open your terminal application and run `$ git --version` which should return the version of your git installation.

Installation Instructions: Mac

To use these instructions, you will need to use your computer's command prompt, an application named *Terminal*. The Terminal application on a Mac is located in the *Utilities* folder, which is located inside the *Applications* folder. Alternatively, you can locate the Terminal application by clicking on the magnifying glass icon at the upper right corner of the screen and typing "Terminal".

We will go much more into detail about how to use the command prompt when we start coding, but for now, simply note that the commands here are preceded by a `$`. When typing the commands yourself, do not type the $ - this is simply a common convention used to indicate that what follows is a command to be run.

To install Git on your personal Mac, follow these steps:

1. Install Homebrew, a package manager for Mac (skip this step if Homebrew is already installed on your computer): Go to the Terminal and copy and paste this code at the prompt:

   ```
   $ /usr/bin/ruby -e "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/master/i
   nstall)"
   ```

   Note: the above command spans one line.

2. Run the following command so that Homebrew packages are run before the system versions of the same (which may be old or not what we want):

   ```
   $ echo 'export PATH=/usr/local/bin:$PATH' >>
   ~/.bash_profile
   ```

3. Install Git with this command: `$ brew install git`

Next, open your terminal application and run `$ git --version` which should return the version of your Git installation.

Creating a Github Account

If you have not already created a personal GitHub account, head over to the GitHub website and sign up for a free account now. Take note of the email address you use, because you are going to use it to connect your computer's Git installation with GitHub.

When creating an account, consider choosing a username that you will feel comfortable sharing in a professional setting. Also, keep in mind that all of the work that you will be adding to your GitHub account is viewable to the public. Make a commitment to always present the best version of your code. Again, your Github profile will serve as your resume and portfolio of coding abilities. Make sure your username is professional, and code within it is clean and presentable.

Once you have created an account on Github, set your personal info by setting global, or computer-wide, variables on your computer for your name and email. Open Terminal, then run the following commands (insert your Github username and email when you do this - these details will be associated with any commits that you create):

```
$ git config --global user.name "first-name second-name"
$ git config --global user.email "email@example.com"
```

Replace `first-name` with your first name and `second-name` with your second name. On the second line, replace `email@example.com` with the email you used to register on GitHub.

Once this is done, the name and email are set for any project located anywhere on the computer. If you run these on your personal computer, you only need to run these commands one time for the names to be set for every project.

Atom

When we write code, we do it in a text editor. Word processing programs like Microsoft Word are not ideal for writing code because they add styles and formatting that can get in the way of our code. We want plain, raw text without any extras for our coding files. If you have a Mac, your device comes with a text editor called TextEdit which can be found in your *Applications* list. If you have a PC, the default text editor is Notepad (Windows) or Gedit (Linux), which can be found though the *Start* menu. The Mac and Linux Terminal applications also include a console-based text editor called Nano. These text editors allow you to write code with plain, raw text.

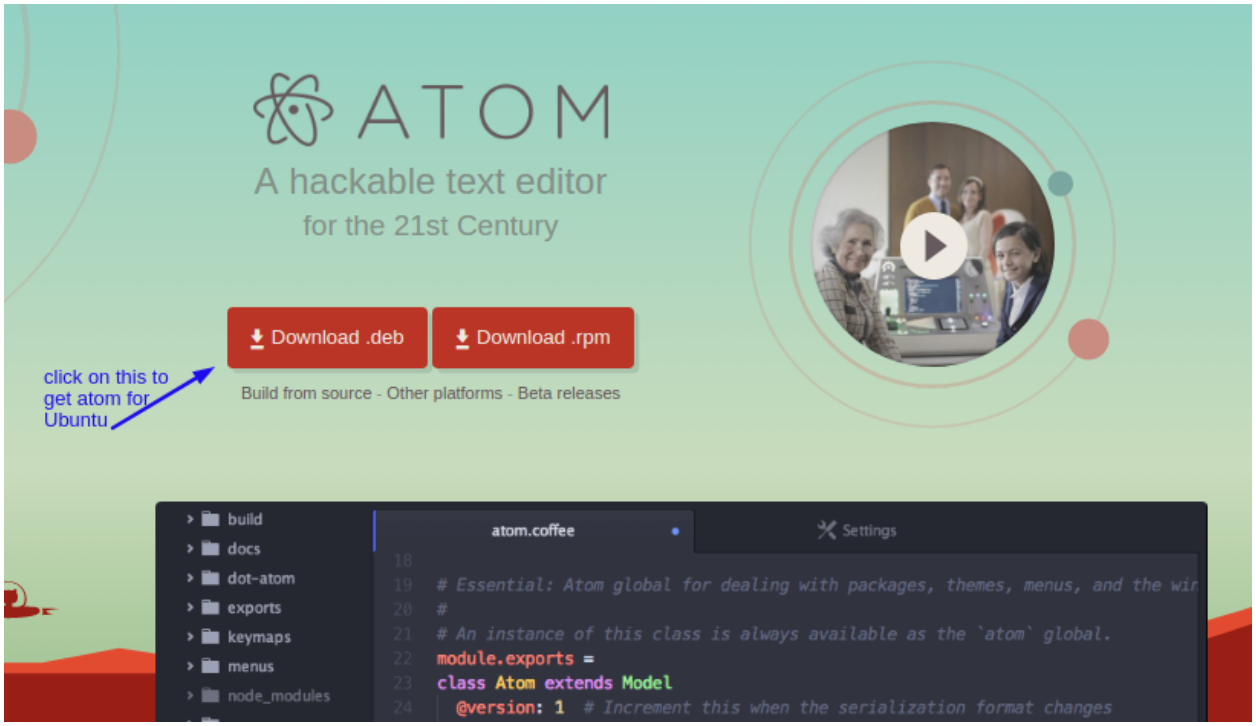
At Moringa School, we use the free text editor, **Atom**. Atom was designed with the developer in mind by adding features and customization that help developers write their code more productively.

During the Moringa Data Science Prep, we will use **Atom** while learning how to work with Git and Github.

Installation

To use Atom on your device, download the appropriate installer for your operating system at the Atom website. Note that for Atom installation to be successful, Git needs to be installed first. Please make sure Git has successfully installed on your computer before proceeding.

Follow the steps in the images and then click on the downloaded content to install Atom Text Editor.



Overview

Let's take a quick tour of Atom. Atom offers the ability to customize the editor's settings and shortcuts. Press *Cmd* + , [*Cmd* + comma] (Mac) or *Ctrl* + , [*Ctrl* + comma] (Linux) to see the *Settings* view.

One setting that we want to make sure we have in place is the tab length. Writing good code means making a habit of using consistent indentation. The standard for indenting lines of code is 2 spaces. We have confirmed that this is set to 2. So, rather than hitting the spacebar twice, we can use the tab key for indentation.

You won't need to worry too much about all of the customization possibilities now but as you start gaining coding experience, remember that you can return to the *Settings* view to make updates to keyboard shortcuts, themes, or packages that add other functionality to the editor.

Slack

At Moringa School, we communicate using a messaging platform named Slack. You will receive an invitation to join the Moringa School Slack group before your class begins via

the email you provided during registration. Please make sure to accept that invitation and log in to the Slack group before class starts. When signing up for an account on Slack, use the same first and last name that you used for your Moringa School registration. Make sure to use your full name.

The Moringa School Slack channel can be accessed on the web at https://moringaschool.slack.com/. A desktop application is also available for download here, but it is not necessary - the application uses a lot of memory and can make your computer run slowly. If possible, please also download the application on your smartphone (available on your operating system's App Store).

Google Group

You should have received an invitation to join your cohort's Google email group before class via the email you provided during registration. Please make sure to accept that invitation through the email now if you have not already done so.